C# 프로그래밍

- 14 주차 (2강)

SEJONG UNIVERSITY

예외 처리
- 예외 필터
- 예외 처리의 사용

SEJONG UNIVERSITY

예외 필터 (Exception Filter)

- C# 6.0부터는 catch 절이 특정 조건을 만족하는 예외 객체를 받도록 지원
  - ✓ when 키워드를 이용해서 제약 조건을 기술

사용 예제:

```
class FilterableException : Exception
{
    public int ErrorNo { get; set; }
}
```

```
static void Main(string[] args)
{
    try
    {
        int num = -1;
        if (num < 0 || num > 10)                                    // num이 0보다 작거나 10보다 큰경우
            throw new FilterableException() { ErrorNo = num }; // FilterableException 예외를 던짐
        else
            Console.WriteLine($"Output : {num}");
    }
    catch (FilterableException e) when (e.ErrorNo < 0) // when 키워드로 ErrorNo이 0보다 작은 경우만,
    {                                                                  // catch 문에서 예외를 받음
        Console.WriteLine("Negative input is not allowed.");
    }
}
```

예외 필터 예제 코드

```
class FilterableException : Exception
{
    public int ErrorNo { get; set; }
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("Enter Number Between 0~10");
    string input = Console.ReadLine();
    try
    {
        int num = Int32.Parse(input);
        if (num < 0 || num > 10)
            throw new FilterableException() { ErrorNo = num };
        else
            Console.WriteLine($"Output : {num}");
    }
    catch (FilterableException e) when (e.ErrorNo < 0)
    {
        Console.WriteLine("Negative input is not allowed.");
    }
    catch(FilterableException e) when (e.ErrorNo > 10)
    {
        Console.WriteLine("Too big number is not allowed.");
    }
}
```

출력 결과:

```
>ExceptionFiltering.exe
Enter Number Between 0~10
5
Output : 5

>ExceptionFiltering.exe
Enter Number Between 0~10
-1
Negative input is not allowed.

>ExceptionFiltering.exe
Enter Number Between 0~10
15
Too big number is not allowed.
```

예외 처리의 사용

- try ~ catch 문을 이용한 예외 처리 효과
  - ✓ 예외를 종류별로 정리하여 코드의 가독성이 향상
  - ✓ try 문에서 발생할 수 있는 다수의 예외 상황을 하나의 catch 문에서 처리 가능
  - ✓ StackTrace 프로퍼티를 통해 예외 발생 위치추적 용이
- try ~ catch 문을 통한 예외 처리의 단점
  - ✓ 기본 예외처리보다 예외 발생 시 처리 시간이 오래 걸림

기본 예외 처리 방식:

```
static int Divide(int dividend, int divisor, out int result)
{ // divisor가 0이면 음수를 반환, 그렇지 않으면 몫을 반환
    if(divisor == 0)
    {
        result = 0;
        return -5;
    }
    else
    {
        result = dividend / divisor;
        return 0;
    }
}
```

try ~ catch 문을 통한 예외 처리 방식:

```
static void Main(string[] args)
{
    try
    {
        int a = 1;
        Console.WriteLine(5 / a);
        Console.WriteLine(3 / --a); // 에외 발생
    }
    catch (DivideByZeroException e)
    {
        Console.WriteLine(e.StackTrace); // 에외 위치추적
    }
}
```

대리자
- 형식으로의 대리자
- 대리자의 활용
- 일반화 대리자

대리자 (delegate)

- 대리자는 메서드에 대한 참조
  - ✓ 참고로, "참조"라는 용어는 객체의 주소를 가리키는 것을 의미
  - ✓ 대리자에 메소드의 주소를 할당한 후 대리자를 호출하면 이 대리자가 메소드를 호출
- 대리자는 delegate 키워드를 이용해서 선언
  - ✓ 클래스를 선언하는 위치와 같은 위치라면 어디든지 선언할 수 있음

사용 형식:

```
한정자 delegate 반환_형식 대리자_이름( 매개변수_목록 );
```

사용 예제:

```
delegate int MyDelegate(int a, int b);
```

형식(type)으로의 대리자

- 대리자는 인스턴스가 아닌 데이터 형식(type)
  - ✓ 메소드를 참조하기 위해 대리자의 인스턴스를 생성하여 사용
  - ✓ 대리자의 인스턴스는 메서드를 참조

대리자 선언 :

```
delegate int MyDelegate(int a, int b);
```

대리자가 참조할 메소드를 선언 :
- 대리자의 반환 형식과 매개변수를 따라야함

```
int Plus(int a, int b)
{
    return a + b;
}

int Minus(int a, int b)
{
    return a - b;
}
```

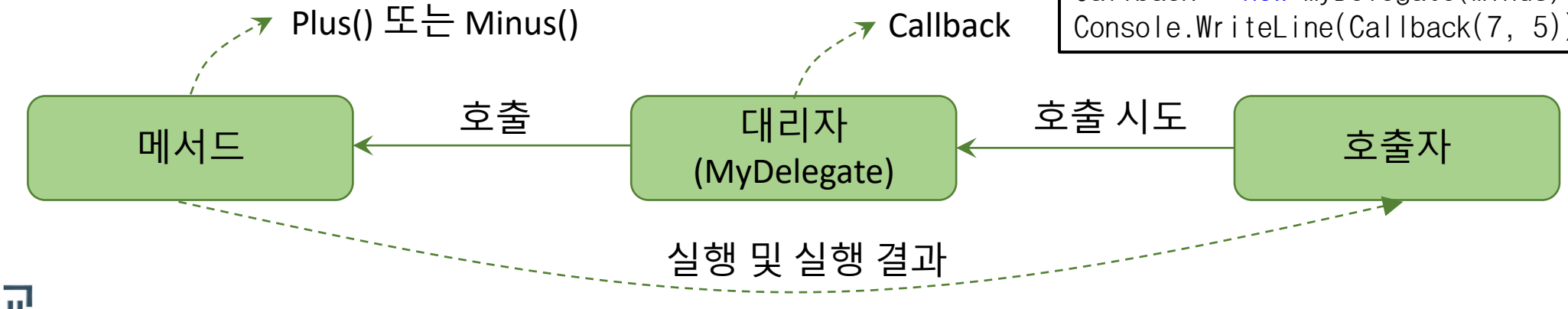
대리자의 인스턴스 생성 :
- 인스턴스는 Plus() 또는 Minus() 메서드 참조

```
MyDelegate Callback;
Callback = new MyDelegate(Plus); // new 연산자를 이용해 Callback 인스턴스 생성
Console.WriteLine(Callback(3, 4)); // 7 출력
Callback = new MyDelegate(Minus);
Console.WriteLine(Callback(7, 5)); // 2 출력
```

대리자 인스턴스 생성부터 호출까지

- 이전 슬라이드의 사용예제를 통한 설명 :
  - ✓ Callback은 반환 형식이 int, 매개변수가 (int, int)인 MyDelegate 대리자 인스턴스
  - ✓ MyDelegate() 생성자 호출해서 Callback 인스턴스 생성
  - ✓ 생성자의 인수로 Plus() 또는 Minus()를 사용하여 Callback이 해당 메서드를 참조하게 만듦
  - ✓ 메서드 호출하듯이 Callback을 사용하면 현재 참조하는 메서드를 실행하여 결과를 반환

```
MyDelegate Callback;
Callback = new MyDelegate(Plus);
Console.WriteLine(Callback(3, 4));
Callback = new MyDelegate(Minus);
Console.WriteLine(Callback(7, 5));
```



대리자 인스턴스 생성 예제 코드

```
delegate int MyDelegate(int a, int b);

class Calculator
{
    public int Plus(int a, int b) // 인스턴스 메서드
    {
        return a + b;
    }

    public static int Minus(int a, int b) // 정적 메서드
    {
        return a - b;
    }
}
```

```
static void Main(string[] args)
{
    Calculator Calc = new Calculator();
    MyDelegate Callback;

    // 인스턴스 메서드 참조
    Callback = new MyDelegate(Calc.Plus);
    Console.WriteLine(Callback(3, 4));

    // 정적 메서드 참조
    Callback = new MyDelegate(Calculator.Minus);
    Console.WriteLine(Callback(7, 5));
}
```

출력 결과:
7
2

대리자의 활용

- 대리자는 메서드를 매개변수로 넘기기 위해 사용
  - ✓ 예를 들어, 배열을 정렬하는 메서드에서 비교 메서드를 매개변수로 받는 경우
  - ✓ 매개변수로 전달받은 비교 메서드에 따라 오름차순 정렬인지 내림차순 정렬인지 결정됨
- 대리자를 사용해 비교 메서드를 매개변수로 받는 버블정렬 메서드 작성

Step 1: 먼저 Compare 대리자를 선언

```
delegate int Compare(int a, int b);
```

Step 2: Compare 대리자가 참조할 비교 메서드를 작성

```
static int AscendCompare(int a, int b)
{
    if (a > b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}
```

대리자의 활용

- 대리자를 사용해 비교 메서드를 매개변수로 받는 버블정렬 메서드 작성

Step 3: 정렬 메서드를 작성. 이때 매개변수로 두 정수를 입력 받아 비교하는 비교 메서드를 참조하는 대리자를 사용

```
static void BubbleSort(int[] DataSet, Compare Comparer)
{
    int i = 0;
    int j = 0;
    int temp = 0;

    for(i=0; i<DataSet.Length-1; i++)
    {
        for(j=0;j<DataSet.Length-(i+1); j++)
        {
            if(Comparer(DataSet[j], DataSet[j+1])>0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

Comparer가 어떤 메서드를 참조하고 있는가에 따라 정렬 결과가 달라짐

대리자의 활용

- 대리자를 사용해 비교 메서드를 매개변수로 받는 버블정렬 메서드 작성

Step 4: 정렬 메서드 BubbleSort() 호출

```
int[] array = { 3, 7, 4, 2 ,10};

Console.WriteLine("Sorting ascending...");
BubbleSort(array, new Compare(AscendCompare)); // array는 {2,3,4,7,10}
```

대리자의 활용 예제 코드

```
delegate int Compare(int a, int b);

static int AscendCompare(int a, int b)
{
    if (a > b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}

static int DescendCompare(int a, int b)
{
    if (a < b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}
```

```
static void BubbleSort(int[] DataSet, Compare Comparer)
{
    int i = 0;
    int j = 0;
    int temp = 0;

    for(i=0; i<DataSet.Length-1; i++)
    {
        for(j=0;j<DataSet.Length-(i+1); j++)
        {
            if(Comparer(DataSet[j], DataSet[j+1])>0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

```
,10};

ascending...");
are(AscendCompare));

Length; i++)
i]} ");

0, 11 };
ng descending...");
pare(DescendCompare));

    for (int i = 0; i < array2.Length; i++)
        Console.Write($"{array2[i]} ");

    Console.WriteLine();
}
```

대리자의 활용 예제 코드

```
delegate int Compare(int a, int b);

static int AscendCompare(int a, int b)
{
    if (a > b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}

static int DescendCompare(int a, int b)
{
    if (a < b)
        return 1;
    else if (a == b)
        return 0;
    else
        return -1;
}
```

```
static void BubbleSort(int[] DataSet, Compare Comp
{
    int i = 0;
    int j = 0;
    int temp = 0;

    for(i=0; i<DataSet.Lengt
    {
        for(j=0;j<DataSet.Le
        {
            if(Comparer(Data
            {
                temp = DataSe
                DataSet[j + 
                DataSet[j] =
            }
        }
    }
}
```

```
static void Main(string[] args)
{
    int[] array = { 3, 7, 4, 2 ,10};

    Console.WriteLine("Sorting ascending...");
    BubbleSort(array, new Compare(AscendCompare));

    for (int i = 0; i < array.Length; i++)
        Console.Write($"{array[i]} ");

    int[] array2 = { 7, 2, 8, 10, 11 };
    Console.WriteLine("₩nSorting descending...");
    BubbleSort(array2, new Compare(DescendCompare));

    for (int i = 0; i < array2.Length; i++)
        Console.Write($"{array2[i]} ");

    Console.WriteLine();
}
```

출력 결과:

```
Sorting ascending...
2 3 4 7 10
Sorting descending...
11 10 8 7 2
```

일반화 대리자

- 대리자는 일반화 메서드도 참조 가능
  - ✓ 대리자도 형식 매개변수를 이용하여 선언 (일반화 대리자)

Step 1 : 대리자를 일반화하여 선언

```
delegate int Compare<T>(T a, T b);
```

Step 2 : 일반화 대리자를 매개변수로 사용하는 버블정렬 메서드 정의

```
static void BubbleSort<T>(T[] DataSet, Compare<T> Comparer)
{
    int i = 0;
    int j = 0;
    T temp;

    for (i = 0; i < DataSet.Length - 1; i++)
    {
        for (j = 0; j < DataSet.Length - (i + 1); j++)
        {
            if (Comparer(DataSet[j], DataSet[j + 1]) > 0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

일반화 대리자

Step 3 : 일반화 대리자가 참조할 일반화 메서드 정의
- 인터페이스 IComparable<T> 를 상속하는 데이터형식 T 를 사용
- IComparable<T>를 상속받은 데이터형식은 CompareTo() 메서드가 구현되어 있음
- CompareTo() 메서드는 매개변수가 자신보다 크면 -1, 같으면 0, 작으면 1을 반환
- int, double 을 포함한 모든 수치 데이터형식과 string 형은 IComparable<T>를 상속 받음

```
static int AscendCompare<T>(T a, T b) where T : IComparable<T>
{
    return a.CompareTo(b);
}
```

Step 4: 일반화 정렬 메서드 BubbleSort<T>() 호출

```
int[] array = { 3, 7, 4, 2, 10 };

Console.WriteLine("Sorting ascending...");
BubbleSort<int>(array, new Compare<int>(AscendCompare));
```

일반화 대리자 예제 코드

```
delegate int Compare<T>(T a, T b);

static int AscendCompare<T>(T a, T b)
where T : IComparable<T>
{
    return a.CompareTo(b);
}

static int DescendCompare<T>(T a, T b)
where T : IComparable<T>
{
    return a.CompareTo(b) * -1;
    // -1을 곱하면 자신보다 큰 경우 1,
    // 작은 경우 -1 반환
}
```

```
static void BubbleSort<T>(T[] DataSet, Compare<T> Comparer)
{
    int i = 0;
    int j = 0;
    T temp;

    for (i = 0; i < DataSet.Length - 1; i++)
    {
        for (j = 0; j < DataSet.Length - (i + 1); j++)
        {
            if (Comparer(DataSet[j], DataSet[j + 1]) > 0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

```
    BubbleSort<string>(array2, new Compare<string>(DescendCompare));

    for (int i = 0; i < array2.Length; i++)
        Console.Write($"{array2[i]} ");
}
```

일반화 대리자 예제 코드

```
delegate int Compare<T>(T a, T b);

static int AscendCompare<T>(T a, T b)
where T : IComparable<T>
{
    return a.CompareTo(b);
}

static int DescendCompare<T>(T a, T b)
where T : IComparable<T>
{
    return a.CompareTo(b) * -1;
    // -1을 곱하면 자신보다 큰 경우 1,
    // 작은 경우 -1 반환
}
```

```
static void BubbleSort<T>(T[] DataSet, Compare<T> Comparer)
{
    int i = 0;
    int j = 0;
    T temp;

    for (i = 0; i < DataSet.Length - 1; i++)
    {
        fo
        {
        }
    }
}
```

```
static void Main(string[] args)
{
    int[] array = { 3, 7, 4, 2, 10 };

    Console.WriteLine("Sorting ascending...");
    BubbleSort<int>(array, new Compare<int>(AscendCompare));

    for (int i = 0; i < array.Length; i++)
        Console.Write($"{array[i]} ");

    string[] array2 = { "abc", "def", "ghi", "jkl", "mno" };
    Console.WriteLine("₩nSorting descending...");
    BubbleSort<string>(array2, new Compare<string>(DescendCompare));

    for (int i = 0; i < array2.Length; i++)
        Console.Write($"{array2[i]} ");
}
```

출력 결과:

```
Sorting ascending...
2 3 4 7 10
Sorting descending...
mno jkl ghi def abc
```

Thank you!